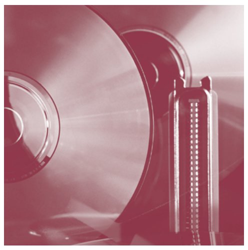
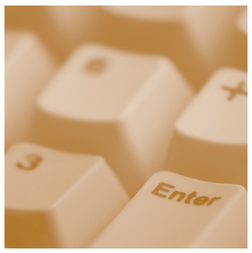
ASG-DataManager™ 2000/80 Interface

Version: 2.5

Publication Number: DMR0200-25-S2K80

Publication Date: June 1981





ASG Worldwide Headquarters Naples, Florida USA | asg.com
1333 Third Avenue South, Naples, Florida 34102 USA Tel: 941.435.2200 Fax: 941.263.3692 Toll Free: 1.800.932.5536

ASG Documentation/Product Enhancement Fax Form

Please FAX comments regarding ASG products and/or documentation to (941) 263-3692.

Company Name	Telephone Number	Site ID	Contact name

Product Name/Publication	Version #	Publication Date
Product:		
Publication:		
Tape VOLSER:		

Enhancement Request:

ASG Support Numbers

ASG provides support throughout the world to resolve questions or problems regarding installation, operation, or use of our products. We provide all levels of support during normal business hours and emergency support during non-business hours. To expedite response time, please follow these procedures.

Please have this information ready:

- Product name, version number, and release number
- List of any fixes currently applied
- Any alphanumeric error codes or messages written precisely or displayed
- A description of the specific steps that immediately preceded the problem
- The severity code (ASG Support uses an escalated severity system to prioritize service to our clients. The severity codes and their meanings are listed below.)

If You Receive a Voice Mail Message:

1. Follow the instructions to report a production-down or critical problem.
2. Leave a detailed message including your name and phone number. A Support representative will be paged and will return your call as soon as possible.
3. Please have the information described above ready for when you are contacted by the Support representative.

Severity Codes and Expected Support Response Times

Severity	Meaning	Expected Support Response Time
1	Production down, critical situation	Within 30 minutes
2	Major component of product disabled	Within 2 hours
3	Problem with the product, but customer has work-around solution	Within 4 hours
4	"How-to" questions and enhancement requests	Within 4 hours

ASG provides software products that run in a number of third-party vendor environments. Support for all non-ASG products is the responsibility of the respective vendor. In the event a vendor discontinues support for a hardware and/or software product, ASG cannot be held responsible for problems arising from the use of that unsupported version.

Business Hours Support

Your Location	Phone	Fax	E-mail
United States and Canada	800.354.3578 1.941.435.2201 **Secondary Numbers:** 800.227.7774 800.525.7775	941.263.2883	support@asg.com
Australia	61.2.9460.0411	61.2.9460.0280	support.au@asg.com
England	44.1727.736305	44.1727.812018	support.uk@asg.com
France	33.141.028590	33.141.028589	support.fr@asg.com
Germany	49.89.45716.300	49.89.45716.400	support.de@asg.com
Singapore	65.224.3080	65.224.8516	support.sg@asg.com
All other countries:	1.941.435.2201		support@asg.com

Non-Business Hours - Emergency Support

Your Location	Phone	Your Location	Phone
United States and Canada	800.354.3578 1.941.435.2201 **Secondary Numbers:** 800.227.7774 800.525.7775 **Fax:** 941.263.2883		
Asia	011.65.224.3080	**Japan/Telecom**	0041.800.9932.5536
Australia	0011.800.9932.5536	**New Zealand**	00.800.9932.5536
Denmark	00.800.9932.5536	**South Korea**	001.800.9932.5536
France	00.800.9932.5536	**Sweden/Telia**	009.800.9932.5536
Germany	00.800.9932.5536	**Switzerland**	00.800.9932.5536
Hong Kong	001.800.9932.5536	**Thailand**	001.800.9932.5536
Ireland	00.800.9932.5536	**United Kingdom**	00.800.9932.5536
Israel/Bezeq	014.800.9932.5536		
Japan/IDC	0061.800.9932.5536	**All other countries**	1.941.435.2201

ASG Web Site

Visit http://www.asg.com, ASG's World Wide Web site.

Submit all product and documentation suggestions to ASG's product management team at http://www.asg.com/products/suggestions.asp

If you do not have access to the web, FAX your suggestions to product management at (941) 263-3692. Please include your name, company, work phone, e-mail ID, and the name of the ASG product you are using. For documentation suggestions include the publication number located on the publication's front cover.

PREFACE

This manual is one of a series describing DATAMANAGER, the data dictionary software developed by MSP for use on IBM System/360, System/370, 303x and 4300 series, and plug compatible, machines. This manual describes the System 2000/80 Interface facility; a facility (additional to the basic set-up, maintenance and interrogation features) which enables the user to include System 2000/80 database data definitions in the data dictionary and to produce System 2000/80 Define Module commands and System 2000/80 Subschema Definitions in COBOL, PL/I or Basic Assembler Language.

This manual supersedes the System 2000 Interface manual, which was first published in October 1979. This manual relates to DATAMANAGER Release 4.0 and later releases, and to System 2000/80 release 2.9.0. The DATAMANAGER System 2000/80 Interface is fully upwards compatible with the former DATAMANAGER System 2000 Interface.

It is assumed that the reader has a knowledge of DATAMANAGER to the extent covered by the User's Guide, and is familiar with System 2000/80.

Chapter 1 of this manual summarises the interface between DATAMANAGER and System 2000/80.

Chapter 2 discusses briefly the concept of System 2000/80 databases and illustrates how System 2000/80 can be defined to DATAMANAGER.

Chapter 3 gives the specifications of the DATAMANAGER data definition statements for System 2000/80 databases.

Chapter 4 describes the interface between System 2000/80 and the DATAMANAGER Source Language Generation facility.

Appendix 1 describes two installation macros that permit the System 2000/80 Interface to be tailored to an installation's own standards.

The notation used in the specification of DATAMANAGER commands is described on page vi.

For the storage and job control requirements for installing and running DATAMANAGER with the System 2000/80 Interface, reference should be made to the Installation in OS Environments manual or Installation in DOS Environments manual, as appropriate.

PREFACE
(continued)

Other manuals in the DATAMANAGER series are the User's Guide; the Messages Manual, in which all diagnostic messages that can be output by DATAMANAGER are listed; the Controller's Manual, circulation of which is restricted to those responsible for the administration of a data dictionary; and a separate manual for each of the separate DATAMANAGER Facilities (Automation of Set-Up, Source Language Generation, User Defined Syntax, various other interface facilities, etc.).

CONTENTS

NOTATION FOR STATEMENT FORMATS

In all manuals relating to DATAMANAGER, the following notation is used in the specification of statement formats (for commands and data definition statements):

- all words printed in capitals are statement identifiers or keywords that must be present in full or truncated form in the circumstances stated in the statement specification. The extent beyond which a word must not be truncated is indicated by underlining of the characters that must be retained.

- all words printed in lower case are variables for which the user must substitute a value consistent with the specification

- material enclosed in square brackets [ ] is an option which may be included or omitted as required

- braces { } indicate that a choice must be made of one of the options enclosed within them

- three full stops ... indicate that the material they immediately follow may be repeated. Where ... immediately follows a closing square bracket or brace, the material that can be repeated is bounded by that square bracket or brace and the corresponding opening square bracket or brace. If material can be repeated only a limited number of times, the repetition permitted is stated in the specification.

- other punctuation marks and symbols must be coded as shown, subject to the implications of any square brackets or braces enclosing them; except that where a single quote, ', is shown, a double quote, ", can alternatively be used, provided that the opening and closing quotes of any pair of quotes are the same character (single quote or double quote).

CHAPTER 1 DATAMANAGER/SYSTEM 2000/80 INTERFACE FACILITIES

For the System 2000/80 user, the following interface facilities between DATAMANAGER and System 2000/80 are provided by MSP:

- the ability to define System 2000/80 databases to DATAMANAGER, to hold the definitions in the data dictionary and to process them by the standard DATAMANAGER commands

- the ability to generate from the data dictionary, and to insert into the required source library, System 2000/80 Define Module commands and System 2000/80 Subschema Definitions in COBOL, PL/I or Basic Assembler Language.

The ability to define a System 2000/80 database demands two further member types in DATAMANAGER. These member types are SYSTEM2000-DATABASE, which in the member type. hierarchy effectively replaces FILE (for System 2000/80 only, as both FILE and SYSTEM2000-DATABASE definitions can exist in the same dictionary), and REPEATING-GROUP (or SCHEMA-RECORD), which in the member type hierarchy comes between SYSTEM2000-DATABASE and GROUP. (The term 'schema record' was introduced in Release 2.9.0 of SYSTEM 2000/80 as a synonym for the earlier term 'repeating group'.) Additionally, facilities at the MODULE, PROGRAM and/or SYSTEM data definition levels are required to allow the processing view of a database to be defined. The SYSTEM2000-DATABASE and REPEATING-GROUP data definition statements and the relevant formats of the MODULE, PROGRAM and SYSTEM data definition statements are further discussed in Chapter 2 and are specified in Chapter 3.

So that the definitions of System 2000/80 databases can be processed by DATAMANAGER in the same way as other members of the data dictionary, the keywords SYSTEM2000-DATABASES, REPEATING-GROUPS and SCHEMA-RECORDS, together with their shorter synonyms S2K-DATABASES, RGS and SRS, are added to the member-type keywords available for use in the following commands:

- BULK
- GLOSSARY
- LIST
- PERFORM
- REPORT
- WHICH

1 (continued)

The ability to generate System 2000/80 Define Module commands and System 2000/80 Subschema Definitions from data dictionary members requires the use of the Source Language Generation facility, which is described in a separate manual. The Source Language Generation manual describes the basic version of the facility, which can output data descriptions in COBOL, PL/I or BAL, and record layouts. The enhancements to the facility to enable it to generate System 2000/80 specific outputs are discussed in Chapter 4 of this manual.

CHAPTER 2 SYSTEM 2000/80 DATABASES AND DATAMANAGER

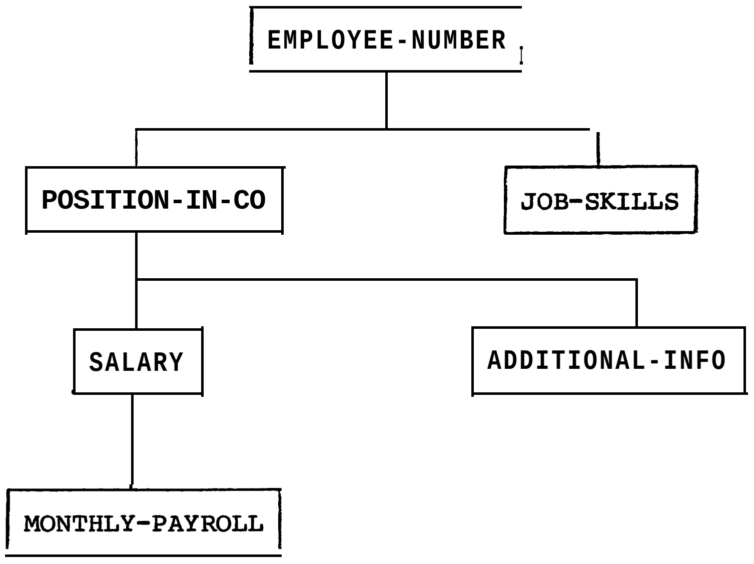
2.1 INTRODUCTION

The ability to give a complete definition of a System 2000/80 database is available within DATAMANAGER.

Conceptually a System 2000/80 database can be viewed with some disregard to its physical organization. This physical organization is, of course, important in the context of storage and processing efficiency but does not concern us here.

A System 2000/80 database can be viewed as a collection of logical entries. A logical entry is a repeating group, or schema record. (These terms are synonymous.) Repeating groups are logically organized into a data tree which is strictly hierarchical (a repeating groupcannot have more than one parent repeating group). The maximum level of depth for a data tree is thirty-two levels. Access to a System 2000/80 database always centres'on a specific repeating group. Within System 2000/80, a database is defined via Define Module Commands which identify and describe each component of the database.

Consider, for example, a database as follows:



2.1 (continued)

EMPLOYEE-NUMBER, POSITION-IN-CO, SALARY, MONTHLY-PAYROLL, ADDITIONAL-INFO and JOB-SKILLS are all repeating groups.

For DATAMANAGER the above database would be represented as a SYSTEM2000-DATABASE member containing six REPEATING-GROUP members. The SYSTEM2000-DATABASE member represents the collection of repeating groups and specifies the data tree into which they are logically organized. The components of the repeating groups are defined via normal DATAMANAGER ITEM and GROUP members.

This then is a simplified look at the database side of the stored data. Still to be defined is the way in which the data is accessed and manipulated. Any given program that accesses the database is unlikely to access every component, so a convenient method of indicating the actual components required is needed. The DATAMANAGER data definition statements MODULE, PROGRAM and/or SYSTEM are used to define the processing view of the database(s) to be accessed. In System 2000/80 terminology this is the PLI (procedural language interface) Subschema Definition.

If certain assumptions are made regarding specific attributes of the database illustrated above, then the following would be the method of using DATAMANAGER data definition statements to define the database. The" data definition shown can be inserted into the data dictionary's source data set by INSERT or ADD commands, in the same way as any other DATAMANAGER data definitions.

```
ADD EMPLOYEE;
SYSTEM2000-DATABASE
CONTAINS EMPLOYEE-NUMBER,
         POSITION-IN-CO     PARENT  EMPLOYEE-NUMBER,
         SALARY             PARENT  POSITION-IN-CO,
         MONTHLY-PAYROLL    PARENT  SALARY,
         ADDITIONAL-INFO    PARENT  POSITION-IN-CO,
         JOB-SKILLS         PARENT  EMPLOYEE-NUMBER
   DATASETS ROLLBACK-LOG DEVICE 3330
DESCRIPTION
'EXAMPLE AS IN INTEL SYSTEM 2000/80 LANGUAGE SPECIFICATION'
'MANUAL'
;
```

2.2 FURTHER CONSIDERATIONS

Although a System 2000/80 specific member type, REPEATING-GROUP, has been provided in DATAMANAGER, to enable users to define a SYSTEM2000-DATABASE as containing repeating groups, DATAMANAGER does also accept the names of ordinary DATAMANAGER GROUP members in the CONTAINS clause of the SYSTEM2000-DATABASE data definition. When the Source Language Generation facility is used to generate System 2000/80 source language statements from a SYSTEM2000-DATABASE member, any such GROUP members whose names appear in the CONTAINS clause are treated as though they were REPEATING-GROUP members. However, any command selecting on REPEATING-GROUPS will not select ordinary GROUP members that are used in this way.

Similarly, as well as accepting the names of ITEM members, DATAMANAGER accepts the names of GROUP members (though not the names of other REPEATING-GROUPS) in the CONTAINS clause of a REPEATING-GROUP member. Because System 2000/80 repeating groups cannot contain any groups, the Source Language Generation facility treats any GROUP members declared in the CONTAINS clause of a REPEATING-GROUP member as composite items when generating Define Module commands; reference paths from 'such GROUP members are not traced, and no-statements are generated for their contained groups and/or items.

DATAMANAGER member names have a maximum permitted length of 32 characters. In System 2000/80, longer names are permitted for databases and their components. If it is desired to use a name longer than 32 characters for any of these entities in the System 2000/80 context, the DATAMANAGER member can be defined with a shorter member name, and the definition can include an ALIAS clause (see User's Guide, section 4.3). The ALIAS clause can state a System 2000/80 specific alias, which can be used instead of the member name whenever Source Language Generation is performed from the member in a System 2000/80 context. Alternatively, for components of a REPEATING-GROUP, a local name can be declared in a KNOWN-AS clause; this local name can be used in place of the contained member's name when Source Language Generation is performed from the containing REPEATING-GROUP.

System 2000/80 allows components to be identified by a component number. The SYSTEM2000-DATABASE member type syntax allows component numbers to be stated. Within DATAMANAGER, these component numbers are meaningful only in the context of the database for which they are defined; they can be used in Source Language Generation for that database (see Chapter 4) but they cannot be used to obtain access to the relevant members of the data dictionary.

CHAPTER 3 DATAMANAGER DATA DEFINITIONS FOR SYSTEM 2000/80 DATABASES

3.1 INTRODUCTION

To enable a System 2000/80 environment to be fully reflected in the data dictionary maintained by DATAMANAGER, the DATAMANAGER System 2000/80 Interface provides:

- two additional member types:
    - **SYSTEM2000-DATABASE,** specified in section 3.2
    - REPEATING-GROUP or SCHEMA-RECORD, specified in section 3.3
- an extension to the **MODULE, PROGRAM** and SYSTEM data definition statements, to reflect the processing view of the database. See section 3.4.

3'.2 THE SYSTEM2000-DATABASE DATA DEFINITION STATEMENT

Format

```
{SYSTEM2000-DATABASE}
{S2K-DATABASE       }

[CONTAINS {repeating-group}
          {group          }
    [,{repeating-group} [COMPONENT-NUMBER component-number]
      {group          }
    PARENT {repeating-group }]...]
           {group           }

[ITEM-COMPONENT-NUMBERS   componerit-number IS {item }
                                               {group}
                          [,component-number IS {item }]...]
                                                {group}

[STRING name _[COMPONENT-NUMBER component-number] 'string'
    [DELIMITER character]
    [ACCESSES {item }[,{item }I....]]...
              {group}  {group}

[FUNCTION name [{DECIMAL}] [COMPONENT-NUMBER component-number]
               [{INTEGER}]
               [{DATE   }]
               [{MONEY  }]
    'string' [DELIMITER character]
             [USES {item }[,{item } 1 ...]]...
                   {group}  {group}

[COMPONENT-LIMIT component-limit]

[COMPONENT-NO component-number IS {MEMBER  } name] ...
                                  {STRING  }
                                  {FUNCTION}

[MASTER-PASSWORD item [version]]

[USER-PASSWORDS item [version] [,item [version]]...]

[COMMBLOCK-NAME name]

[DATASETS dataset [dataset]...]

[common clauses]

{;}
{.}
```

3.2
(continued)

where

repeating-group is the name of a member that is a REPEATING-GROUP (or RG or SCHEMA-RECORD or SR)

group is the name of a member that is a GROUP

component-number is an unsigned integer in the range 1 to 4095 (unless the range has been reduced by the tailoring of the installation macro DDS2K: see Appendix 1, section App.1.2)

item is the name of a member that is an ITEM

name is a name conforming to the rules for member names

string is a character string of not more than 250 printable characters. Spaces are regarded as printable characters.

character is an undelimited character string of one character. The character must be one of those specified in the following table:

Valid Delimiter Characters

Character	Hex Value	Character	Hex Value	Character	Hex Value
¢	4A	$	5B	_	6D
.	4B	*	5c		6E
	4c	)	5D	?	6F
(	4D	;	5E	#	7B
+	4E	-	5F	@	7c
|	4F	-	60	'	7D
&	50	/	61	=	7E
!	5A	%	6C	"	7F

component-limit is an unsigned integer in the range 1 to 1000

version is an unsigned integer in the range 1 to 15, being a number specifying which version of the item is relevant to this definition. The version is within the HELD-AS form. If version is omitted, a default value of 1 is assumed.

3.2 (continued)

dataset is:

```
{{UPDATE-LOG  } DEVICE {number}
{{ROLLBACK-LOG}        {'type'}
{{ID-TABLE                    } DEVICE {number} [PAGE-SIZE size]
{{HIERARCHY-TABLE}            }        {'type'}
{{HT             }            }
{{UNIQUE-VALUES-TABLE}        }
{{UVT                }        }
{{MULTIPLE-VALUES-TABLE}      }
{{MVT                  }      }
{{DATA-TABLE}                 }
{{DT        }                 }
{{OVERFLOW-TABLE}             }
{{OT            }             }
```

where

number is an unsigned integer, being a manufacturer's type number. No validation is performed on number.

type is a character string

size is an unsignedinteger, being the size in bytes of the page to be used'for buffering. No validation is performed on size

common clauses are any of the following clauses, as defined in section 4.3 of the User's Guide, in any order:

- ACCESS-AUTHORITY	- FREQUENCY
- ADMINISTRATIVE-DATA	- NOTE'
- ALIAS	- OBSOLETE-DATE
- CATALOGUE	- QUERY
- COMMENT	- SECURITY-CLASSIFICATION
- DESCRIPTION	- SEE
- EFFECTIVE-DATE	

Remarks

1 The member type identifiers SYSTEM2000-DATABASE and S2K-DATABASE are synonymous.

2 Members whose names appear in the CONTAINS clause can be defined as REPEATING-GROUP members or as GROUP members. If the Source Language Generation facility is used to produce System 2000/80 source language statements from this data definition, any directly contained GROUP member is treated as though it had been defined as a REPEATING-GROUP.

3 The CONTAINS clause lists the names of from one to 255 repeating groups that constitute the database. Each name except the first in the list must be preceded by a comma;

3.2
(continued)

the comma can optionally be preceded and/or followed by spaces. The names must be listed in the top to bottom, left to right hierarchical order of the-repeating groups within the database.

4 The first name listed in the CONTAINS clause must be the name of the root or level 0 repeating group.

5 The second and subsequent repeating group names listed must each have an associated PARENT clause, stating the name of the parent repeating group. The name of-the parent repeating group must have appeared earlier in the list of repeating group names.

6 The second and each subsequent repeating group name in the CONTAINS clause can optionally be immediately followed by a COMPONENT-NUMBER clause, stating the System 2000/80 component number of the repeating group within the database. (The first repeating group named in the CONTAINS clause is always component number 0.)

7 Component numbers can be utilized by the PRODUCE command of the Source Language Generation facility, but otherwise they have no significance to DATAMANAGER; they do not , give rise to any indexing.or relationships within the data dictionary.

8 Component numbers can also be defined, in the optional ITEM-COMPONENT-NUMBERS clause, for items and/or groups that are components of the database. Each of these component numbers except the first in the clause must be preceded by a comma; the comma can' optionally be preceded and/or followed by spaces. An item name or group name must not be repeated within the ITEM-COMPONENT-NUMBERS clause.

9 The optional STRING clauses can be used to identify System 2000/80 natural language strings. A component number can optionally be defined for each string. See also remark 11. The Source Language Generation facility passes the string to System 2000/80 exactly as it is defined; DATAMANAGER performs no validation on the string. The optional ACCESSES subordinate clause enables uses/used by relationships to be established by DATAMANAGER between the database and items or groups used in System 2000/80 string operations.

10 The optional FUNCTION clauses can be used to identify System 2000/80 natural language User Functions. One of the function types DECIMAL, INTEGER, DATE or MONEY can optionally be specified for each User Function: if no function type keyword is stated, the System 2000/80 precompiler will assume DECIMAL. A component number can optionally be defined for each User Function. See also

3.2 (continued)

remark 11. The Source Language Generation facility passes the User Function string to System 2000/80 exactly as defined; DATAMANAGER performs no validation on it. The optional USES subordinate clause enables uses/used by relationships to be established by DATAMANAGER between the database and items and groups used in System 2000/80 function operations.

11 For each string defined in a STRING clause or in a FUNCTION clause, the string's delimiter character can also be defined. For any string for which the DELIMITER subordinate clause is omitted, the default character specified by the DELIM paramater of the DGS2K installation macro (see Appendix 1, section App.1.3) is assumed.

12 The optional COMPONENT-LIMIT clause specifies the maximum number of components that can be specified for the database. If the COMPONENT-LIMIT clause is omitted, a default component-limit of 1000 is assumed, unless a lower component-limit has been specified by the NCMAX parameter of the DDS2K installation macro (see Appendix 1, section App.1.2).

13 The optional COMPONENT-NO component-number IS MEMBER name clauses can be used to define the component number of REPEATING-GROUPs, GROUPs or ITEMs within the database, provided that:

- if name is the name of a REPEATING-GROUP, it has not a component number defined for it by a COMPONENT-NUMBER subordinate clause-in the CONTAINS clause

- if name is the name of a GROUP, it has not a component number defined for it by a COMPONENT-NUMBER subordinate clause in the CONTAINS clause, or by the ITEM-COMPONENT-NUMBERS clause

- if name is the name of an ITEM, it has not a component number defined for it by the ITEM-COMPONENT-NUMBERS clause.

If the name declared in one of these clauses is not the name of an existing member of the data dictionary, then when the SYSTEM2000-DATABASE member is encoded, DATAMANAGER generates a dummy ITEM data entries record under that member name.

If the name declared in one of these clauses is the name of an encoded member of the data dictionary and the member is neither a REPEATING-GROUP nor a GROUP nor an ITEM, the clause is rejected with the error message:

```
member-type member-name IS NOT A VALID MEMBER
TYPE IN THIS CONTEXT
```

14 The optional COMPONENT-NO component-number IS STRING name clauses can be used to define the component numbers of natural language strings whose names are declared in STRING clauses, provided that the component numbers for those strings are not defined in COMPONENT-NUMBER subordinate clauses in the STRING clauses.

If the name declared in one of these clauses is not the name of a string declared in a STRING clause, or if the string already has a valid component number declared for it in the STRING clause, the clause is rejected with the error message:

```
name IS AN INVALID COMPONENT-NO CLAUSE
```

15 The optional COMPONENT-NO component-number IS FUNCTION name clauses can be used to define the component numbers of natural language User Functions whose names are declared in FUNCTION clauses, provided that the component numbers of those functions are not defined in COMPONENT-NUMBER subordinate clauses in the FUNCTION clauses.

If the name declared in one of these clauses is not the name of a User Function declared in a FUNCTION clause) or if the User Function already has a valid component number declared for it in the FUNCTION clause, the clause is rejected with the error message:

```
name IS AN INVALID COMPONENT-NO CLAUSE
```

16 When the PRODUCE command of the Source Language Generation facility is used to generate Define Module commands or Subschema Definitions from the SYSTEM2000-DATABASE definition,' then, subject to the conditions stated in the command's specifications in section 4.2,

```
COMPONENT-NO  component-number  IS  {MEMBER  }  name
                                    {STRING  }
                                    {FUNCTION}
DMR-NOTE 'GENERATED BY DATAMANAGER AT hh.mm ON dd mm  yy'
```

I

clauses can be generated and added to the end of the SYSTEM2000-DATABASE definition, for each directly contained member, natural language string or User Function for which a component number is not already specified in the data definition. The SYSTEM 2000-DATABASE member is then re-encoded. This ensures that consistency is maintained between Define Module command and Subschema Definition generations.

17 String names and User Function names are not recorded in the index data set.

3.2
(continued)

18 **DATAMANAGER** checks that:

- no component number is outside the range 1 to 4095

- no component number for a member is outside the range specified by the installed values of the ECMAX and ECMIN parameters of the macro DDS2K

- no component number for a string is outside the range specified by the installed values of the SCMAX and SCMIN parameters of the macro DDS2K

- no component number for a function is outside the range specified by the installed values of the FCMAX and FCMIN parameters of the macro DDS2K

- no component number is duplicated within the database's data definition statement.

The macro DDS2K is documented in Appendix 1, Section App.1.2.

19 The optional MASTER-PASSWORD and USER-PASSWORDS clauses specify that the indicated versions of the HELD-AS form of the stated items have CONTENT IS definitions that specify the passwords to be used.

20 The optional DATASETS clause can be used to specify the devices on which the physical files constituting the database are held, together, where appropriate, with the relevant buffer page size.

21 The database keys and expansion rules are defined in the contained REPEATING-GROUP members.

22 Common clauses, listed under Format above, can be present in any type of data definition statement; they are therefore defined separately, in section 4.3 of the User's Guide. Not more than one of each of these clauses can be declared. If a common clause has a subordinate clause or keyword, the subordinate clause identifier or subordinate keyword must not be truncated to an extent where it becomes ambiguous with any other clause identifier or other keyword available in the data definition syntax for this member type.

23 Clauses can be declared in any order, provided that subordinate clauses are not separated from the other elements of the clause of which they form a part. (For example, the CONTAINS clause includes its subordinate COMPONENT-NUMBER and PARENT clauses, and no other clause must be interposed between its subordinate clauses.)

3.2
(continued)

24 A record containing the database's data definition statement can be inserted into the data dictionary's source data set by a suitable command (see User's Guide, Chapter 3) and an encoded record can subsequently be generated and inserted into the data entries data set. If, when the encoded record is generated, any member whose name appears in the database's data definition statement has no data entries record, DATAMANAGER creates a dummy data entries record for that member; as a dummy REPEATING-GROUP record if the member's name appears in the CONTAINS clause (including its subordinate PARENT clauses), or as a dummy ITEM record if the member's name appears in any other clause.

Example

An example is given in Chapter 2.

3.3 THE REPEATING-GROUP DATA DEFINITION STATEMENT

Format

```
{REPEATING-GROUP}
{RG             }
{SCHFMA-RECORD  }
{SR             }

[{ENTERED-AS  }] [{ALIGNED    }] [CONTAINS content
[{HELD-AS     }] [{UNALIGNED  }]
[{REPORTED-AS }] [{NOT-ALIGNED}]
[{DEFAULTED-AS}]

[{ELSE content [IF clause]} [ELSE content [IF clause]]...
[{IF clause               }                                 ]

[,content [{ELSE content [IF clause]}
          [{IF   clause             }

          [ELSE content [IF clause]]... ]...]
                                      I
KEYS {item-name } [{UNIQUE    }]
     {group-name} [{DUPLICATED}]

     [WITH {NO  } [FUTURE] [{ADDITIONS  }]]
           {FEW }          [{OCCURRENCES}]
           {SOME}
           {MANY}

    [,{item-name } [{UNIQUE    }]
      {group-name} [{DUPLICATED}]

      [WITH {NO  } [FUTURE] [{ADDITIONS  }]11 ·
            {FEW }          [{OCCURRENCES}]
            {SOME}
            {MANY}

[USER-EXIT module-name] [common clauses]

{;}
{.}
```

where

content IF clause module-name	are as defined for a GROUP **data definition** statement in section **4.2 of the User's** Guide

item-name is the name of a member that is an ITEM

group-name is the name of a member that **is a GROUP**

3.3 (continued)

common clauses are any of the following clauses, as defined in section 4.3 of the User's Guide, in any order:

- ACCESS'AUTHORITY
- ADMINISTRATIVE-DATA
- ALIAS
- CATALOGUE
- COMMENT
- DESCRIPTION
- EFFECTIVE-DATE
- FREQUENCY
- NOTE
- OBSOLETE-DATE
- QUERY
- SECURITY-CLASSIFICATION
- SEE

Remarks

1 The member type identifiers REPEATING-GROUP, RG, SCHEMA-RECORD and SR are synonymous.

2 With the exception of the KEYS clause, the format of the REPEATING-GROUP data definition statement following the member type identifier is the same as that of the GROUP data definition statement. In the KEYS clause:

- the keywords ASCENDING and DESCENDING included in the GROUP syntax, are not included for REPEATING-GROUP

- an additional subordinate clause, the WITH clause, is included in the syntax for REPEATING-GROUP.

The specification of the additional subordinate clause is given below; for the specification of the rest of the data definition statement, reference should be made to that of the GROUP statement in section 4.2 of the User's Guide.

3 A WITH subordinate clause can optionally be associated with each item-name declared in the KEYS clause. It is used by the Source Language Generation facility to determine the expansion allowance, thus:

- WITH NO indicates 0% expansion allowance
- WITH FEW indicates 50% expansion allowance
- WITH SOME indicates 100% expansion allowance
- WITH MANY indicates 200% expansion allowance.

The optional keywords FUTURE ADDITIONS and FUTURE OCCURRENCES are for natural language readability and have no processing significance.

4 The terms SCHEMA-RECORD and FUTURE OCCURRENCES were introduced in System 2000/80 Release 2.9.0. The earlier terms were REPEATING-GROUP and FUTURE ADDITIONS respectively.

5 Common clauses, listed under Format above, can be present in any type of data definition statement; they are therefore defined separately, in section 4.3 of the User's Guide. Not more than one of each of these clauses can be declared. If a common clause has a subordinate clause or keyword, the subordinate clause identifier or subordinate keyword must not be truncated to an extent where it becomes ambiguous'with any other clause identifier or other keyword available in the data definition syntax for this member type.

3.4 SYSTEM, PROGRAM AND MODULE DATA DEFINITION STATEMENTS FOR A SYSTEM 2000/80 ENVIRONMENT

3.4.1 Introduction

The data definition statements for DATAMANAGER SYSTEM, PROGRAM and MODULE members acting on conventional files are described in the DATAMANAGER User's Guide. For the System 2000/80 Interface, a further clause, the PROCESSES clause, is included in the format of these statements, to define the processing view of the database. The PROCESSES clause is defined in section 3.4.2. For a full specification of the SYSTEM, PROGRAM and MODULE data definition statements in a System 2000/80 environment, therefore, section 3.4.2 must be read in conjunction with section 4.2 of the User's Guide.

3.4.2 Specification of the PROCESSES Clause

This clause is being re-specified. Its syntax will be issued later.

CHAPTER 4 SYSTEM 2000/80 SOURCE LANGUAGE GENERATION FROM DATAMANAGER

4.1 INTRODUCTION

The DATAMANAGER Source Language Generation facility can be used to produce System 2000/80 source statements of the following types from encoded data definitions held in a DATAMANAGER data dictionary:

- Define Module commands
- Subschema Definitions (with optional COMMBLOCKS) in COBOL, PL/I or Assembler.

Generation of these statements is achieved by use of the PRODUCE command. The basic form of the command, which can generate record layouts or COBOL, PL/I or Assembler data descriptions for conventional file environments, is described in the separate DATAMANAGER manual entitled Source-Language Generation. Users should refer to that manual for a general description of source language generation and of the PRODUCE command. The variations of the command to produce System 2000/80 source statements are described in sections 4.2 and 4.3 below.

An installation macro, DGS2K, permits the output from these variations of the PRODUCE command to be tailored to conform to the particular installation's standards. DGS2K is described in Appendix 1, section App.1.3. Certain parts of the output generated when producing Subschema Definitions can be further tailored by means of the DATAMANAGER installation macros:

- DGCOB, if the data descriptions are produced in COBOL
- DGPLI, if the data descriptions are produced in PL/I
- DGBAL, if the data descriptions are produced in Assembler.

These macros are documented in the Source Language Generation manual.

In the specifications in this chapter, any MSP-defined conditions or values that can be tailored by the Controller are annotated "(unless tailored, see xxxx)", where xxxx is the relevant keyword of the appropriate macro, DGS2K, DGCOB, DGPLI or DGBAL.

4.2 SPECIFICATION OF THE PRODUCE COMMAND FOR SYSTEM 2000/80 DEFINE MODULE COMMANDS

Format

```
{SYSTEM2000}
{S2K       }

DEFINE [COMPONENTS [ONLY {item-name } [,{item-name }]...]]
                         {group-name}    {group-name}

       [FUNCTIONS [ONLY function-name [,function-name]...]]

       [STRINGS [ONLY string-name [,string-name]...]]

FROM member-name [AS library-name]
    [,member-name [AS library-name]]...

[control-options] {;}
                  {.}
```

where

item-name-is the name of a member that is an ITEM.

group-name is the name of a member that is a GROUP or a REPEATING-GROUP or a SCHEMA-RECORD

function-name is a name that appears in a FUNCTION clause of member-name' s data definition

string-name is a name that appears in a STRING clause of member-name's data definition

member-name is the name of an encoded member that is a SYSTEM2000-DATABASE

library-name is the name to be given to the generated library member in the output data set. The name must not be more than 8 characters (unless tailored, see MEMLEN). The first character must be alphabetic or #, £ (or local currency symbol with the internal code hexadecimal 5B), % or @.

control-options is asdefined in Chapter 2 of the Source Language Generation manual, except that the GIVING NOTES and GIVING DESCRIPTIONS clauses there defined are excluded, and either the GIVING clause or the OMITTING clause can optionally include the keyword UPDATES.

Remarks

1 The first elements in the command must be:

- the command identifier, PRODUCE
- the context keyword, SYSTEM2000 or S2K. (The keywords SYSTEM2000 and S2K are synonymous.)
- the DEFINE keyword or a DEFINE clause
- the FROM clause

in that order. Any control options can follow in any order.

2 Up to a maximum of 16 member names can be declared in the FROM clause. If two or more are declared, each except the first must be preceded by a comma; the comma can optionally be preceded and/or followed by spaces.. Member names are processed in the order in which they appear in the FROM clause.

3 Acceptance of the PRODUCE command is in respect of each member-name individually. If member-name:

- is not encoded, or
- is not present in the data dictionary, or
- is not a SYSTEM2000-DATABASE member, or
- is protected against access by the user (see remark 4)

a message is output, no generation takes place in respect of that member-name, and processing continues with the next member-name or command.

4 Acceptance of the PRODUCE command is subject to access security levels (for each member-name individually, as stated in remark 3). See the specification of the PROTECT command in the User's Guide, and the description of the security system in section 1.7 of the User's Guide. If a member-name has an access security level higher than the user's (general or specific) security level, the command is rejected in respect of that .member-name. If the. command can be accepted in respect of member-name, but a reference path from member-name includes a protected member with an access level higher than .the user's security level, the reference path is, if appropriate, followed to its end to determine the total storage space required, but the name of no member in that reference path beyond the last member to which the user has access is given; instead, a filler name is generated.

4.2 (continued)

5 AS clauses are relevant only if System 2000/80 source language statements are being produced and written to an output data set. (System 2000/80 source statements can be generated for listing on a printer or terminal only, without being written to an output data set: see the control options specifications.)

6 Each AS clause present in the command relates only to the member-name that immediately precedes it. It declares a name under which the generated source language data description is to be catalogued in the output source library data set.

7 For each member-name for which no AS clause is specified, library-name is defaulted to member-name if member-name conforms to the length restriction on library-name. The length restriction on library-name is a maximum of eight characters (unless tailored, see MEMLEN). If member-name is longer than the permitted maximum length for library-name, no generation takes place in respect of that member-name, a message is output, and processing continues with the next member-name or command.

8 Library-names, whether declared or defaulted, are not subject to any name editing, ALIAS or WITH-ALIAS clauses (see control option&that may be present in the command.

9 If the keyword DEFINE is specified with no subordinate clauses or keywords, then DEFINE declarations are generated for:

- all repeating groups/schema records, groups and items contained by each accepted member-name specified in the FROM clause
- all functions and strings specified in the data definition of each such member-name

The generation will be successful only if the CONTAINS clause, a STRING clause and a FUNCTION clause are all present in the member-name's definition.

10 If, within the DEFINE clause:

- the keyword COMPONENTS is specified with no ONLY clause, then DEFINE declarations are generated for all repeating groups/schema records, groups and items

- the keyword FUNCTIONS is specified with no ONLY clause, then DEFINE declarations are generated for all functions

- the keyword STRINGS is specified with no ONLY clause, then DEFINE declarations are generated for all strings.

4.2 (continued)

Any combination of these keywords can be specified, in any order, in the DEFINE clause. If any of these keywords is specified, then no generation takes place in respect of any of these **keywords** that is not specified.

11 If, within the **DEFINE** clause, any of the keywords COMPONENTS, FUNCTIONS or STRINGS is specified with an ONLY clause, then the generation in respect of that keyword as defined in remark 10 is restricted to DEFINE declarations for those entities named in the ONLY clause.

12 If any repeating group/schema record, group, item, string or function for which a **DEFINE** declaration is generated does not have a component number specified in the **SYSTEM2000-DATABASE** definition, then **DATAMANAGER generates a** component number for it. Start component numbers and respective increments are determined by the installation macro **DGS2K:** see Appendix 1. The component number of a repeating group/schema record **is** used as the basis for the generation of component numbers for its constituent elements.

13 If:

- the **PRODUCE** command is issued in a non-frozen status, and
- the user has sufficient authority to alter and to re-encode member-name, and
- the value of the UPDATES parameter of the **macro DGS2K** is YES and the PRODUCE command does not include OMITTING UPDATES, or the **PRODUCE** command includes **GIVING UPDATES**

then, subject to remark 15, for each component number generated by DATAMANAGER, DATAMANAGER inserts:

$$\textbf{COMPONENT-NUMBER}\ \text{component-number IS} \begin{Bmatrix} \text{MEMBER} \\ \textbf{STRING} \\ \textbf{FUNCTION} \end{Bmatrix} \text{name}$$

DMR-NOTE 'GENERATED BY DATAMANAGER AT hh.mm ON dd mm yy'

at **the end of the SYSTEM 2000-DATABASE source record. The member is** then re-encoded. This ensures that consistency is maintained between Define Module command **and Subschema Definition generations.**

14 **GIVING UPDATES and OMITTING UPDATES clauses must not both be stated in the same PRODUCE command. These clauses override respectively the NO and the YES values of the UPDATES** parameter of the **DGS2K** macro.

4.2
(continued)

15 If DATAMANAGER attempts to update member-name as stated in remark 13, but either member-name or any other member named in the FROM clause for which a similar attempt is made fails to re-encode, then the updating of all such members is backed out; so that if re-encoding fails for any member named in the FROM clause, neither the data entries nor the source record of any member is left updated by the PRODUCE command. If updating is desired, therefore, it is recommended that a separate PRODUCE command be issued for each member from which Define Module command generation is required.

16 The system separator character is determined by the SEP parameter of the installation macro DGS2K.

17 The System 2000/80 terminology to be generated is determined by the TERM parameter of the installation macro DGS2K.

18 After all editing specified in editing clauses of the control options has been completed, DATAMANAGER performs a final automatic editing of generated data names to ensure that they conform to System 2000/80 rules. Any names longer than the maximum permitted are shortened by removing middle characters.

Example

For an example of the PRODUCE command and the resulting output, see Chapter 7 of the DATAMANAGER Example Book.

4.3 SPECIFICATION OF THE PRODUCE COMMAND FOR SUBSCHEMA DEFINITIONS

This section will be issued later.

CHAPTER 5 SYSTEM 2000/80/DATAMANAGER CORRESPONDENCE TABLES

5.1 INTRODUCTION

This chapter contains tables which indicate the direct relationships that exist between DATAMANAGER data definitions and System 2000/80 source statements.

References in the tables to PRODUCE indicate the Sources Language Generation facility (see Chapter 4).

5.2 DEFINE MODULE COMMANDS RELATIONSHIP TABLE

Correspondence Between System 2000/80 Define Module Commands and DATAMANAGER Data Definitions

System 2000/80 Define Module Syntax	DATAMANAGER Syntax
component number	Either COMPONENT-NUMBER component-number ITEM-COMPONENT-NUMBER component-number COMPONENT-NO' component-number or generated by PRODUCE.
system separator	Value of SEP parameter in DGS2K macro.
component name	DATAMANAGER repeating group/ schema record, group, item, string or function name.

(continued)

(continued)

Correspondence Between System 2000/80 Define Module Commands and DATAMANAGER Data Definitions

System 2000/80 Define Module Syntax	DATAMANAGER Syntax
repeating groups/schema records	Generated from members named in SYSTEM2000-DATABASE CONTAINS list.
SR IN component number	Component number of schema parent except where direct descendant of root.
Elements	Generated for groups and items (groups are treated as composite items and contained members ignored).
KEY/NON-KEY	Determined by KEYS clause in SCHEMA-RECORD definition.
data type NAME	Item defined as ALPHABETIC, ALPHANUMERIC or CHARACTER with COMPRESSED specified.
TEXT	As for NAME but with COMPRESSED not specified.
INTEGER	Item defined as DECIMAL with no decimal places.
DATE	USAGE DATE clause in item definition.
DECIMAL	Item defined as DECIMAL with decimal places specified.
"MONEY	USAGE MONEY clause in item definition
picture	PICTURE clause or determined from the definition.
schema record relationship IN component number	Component number of containing SCHEMA-RECORD, except when contained by root.

(continued)

(continued)

Correspondence Between System 2000/80 Define Module Commands and DATAMANAGER Data Definitions

System 2000/80 Define Module Syntax	DATAMANAGER Syntax
padding option .	KEYS clause in SCHEMA-RECORD definition.
WITH $\left\{\begin{array}{l}\text{NO}\\\text{FEW}\\\text{SOME}\\\text{MANY}\end{array}\right\}$ FUTURE OCCURRENCES	WITH $\left\{\begin{array}{l}\text{NO}\\\text{FEW}\\\text{SOME}\\\text{MANY}\end{array}\right\}$ FUTURE OCCURRENCES
strings	Generated for STRING clauses specified in SYSTEM2000-DATABASE definition.
STRING	STRING
marker	DELIMITER character or value of DLIM parameter in DGS2K macro.
string definition	Character string from STRING clause.
marker	DELIMITER character or value of DLIM parameter in DGS2K macro.
functions	Generated for FUNCTION clauses specified in the SYSTEM2000-DATABASE definition.
function type	DECIMAL INTEGER DATE MONEY
FUNCTION	FUNCTION
marker	DELIMITER character or value of DLIM parameter in DGS2K macro.

(continued)

(continued)

Correspondence Between System 2000/80 Define Module Commands and DATAMANAGER Data Definitions	
System 2000/80 Define Module Syntax	DATAMANAGER Syntax
function definition	Character stying from FUNCTION clause
marker	DLIMITER character or value of DLIM parameter in DGS2K macro.

Tables for System 2000/80 Subschema Definitions and COMMBLOCKS will be issued later.

APPENDIX1 INSTALLATION MACROS

App.1.1 INTRODUCTION

The installation macros DDS2K and DGS2K allow the Controller to tailor the DATAMANAGER System 2000/80 Interface to suit a particular organization's requirements.

DDS2K can be used to tailor the component number ranges and delimiter characters acceptable during the encoding of S2K-DATABASE members.

DGS2K can be used to tailor the generation of System 2000/80 Define Module commands.

The macros are supplied as source modules on the installation magnetic tape. If the supplied default values of all their keywords are acceptable, no further action need be taken in respect of the macros. If any values are to be changed, the procedure described in Chapter 2 of the Installation in OS Environments manual or the Installation in DOS Environments manual, as appropriate, must be followed.

App.1.2 THE MACRO DDS2K

The keywords of the macro DDS2K are defined in the following table. The macro assembles as module DYD99.

The Macro DDS2K: Keywords Specifiable on Installation			
Keyword	**Specifies**	**Default Value**	**Alternative Values**
ECMAX	The maximum component number for an element (member)	4095	1 to 4094
ECMIN	The minimum component number for an element (member)	1	2 to 4095
FCMAX	The maximum component number for a function	4095	1 to 4094
FCMIN	The minimum component number for a function	1	2 to 4095
SCMAX	The maximum component number for a string	4095	1 to 4094
SCMIN	The minimum component number for a string	1	2 to 4095
NCMAX	The default maximum number of components for a database, if no COMPONENT-LIMIT clause is specified in the S2K-DATABASE definition	1000	1 to 999
DLIM MM	The valid System 2000/80 delimiters	All valid delimiters, expressed as a sublist, with each delimiter within quotes	Part of the the default sublist

App.1.3 THE MACRO DGS2K

The keywords of the macro DGS2K are defined in the following table. The macro assembles as module DYL11.

The Macro DGS2K: Keywords Specifiable on Installation			
Keyword	Specifies	Default Value	Alternative Values
ACSMETH	The type of file to be generated by a PRODUCE command	BPAM	QSAM
ALIAS	Whether S2K specific aliases are to be generated instead of member names	NO	YES
AUTOCHK	Check for and convert fillers	YES	NO
CINC	The increment used to calculate component numbers for elements	1	Any valid number (see note 1)
COL01	Column number in which S2K Define Component Declarations begin	1	An integer in the range 2 to 80
CONCARD	Whether a control card is to be generated	YES	NO
DDNAME	File name	'GENLIB'	A delimited string of 1 to 8 characters
DLIM	The default delimiter character to be generated in STRING or FUNCTION declarations, where no delimiter is specified for a STRING or FUNCTION in the DATAMANAGER S2K-DATABASE definition	'*'	Any valid S2K delimiter within quotes
FCINC	The increment used to calculate component numbers for FUNCTIONS	1	Any valid number (see note 1)

(continued)

The Macro DGS2K: Keywords Specifiable on Installation

Keyword	Specifies	Default Value	Alternative Values
FCN	The starting component number for FUNCTIONS	no default	Any valid number (see note 1)
LIBCC	The format of the control card output as the first card of a QSAM file (unless overridden by 'control card* in an ONTO clause) ,	(see section 2.2 of the Source Language Generation manual)	A delimited character string of 1 to 72 characters including question-mark (?)
KNOWN-AS	Whether local names from KNOWN-AS clauses are to be generated instead of member names	NO	YES
MAXSYM	Number of PICTURE symbols at which replacement with a repetition factor takes place (for example, X(3) instead of XXX)	3	1 to 250
MEMLEN	Maximum length of library-name	8	Up to 16
RNDBIN	Whether or not a binary field is to be rounded up to a halfword/fullword	YES (see note 2)	NO
RGI	The increment used to calculate component numbers 'for repeating groups/schema records, the ENTRY repeating group having a component number of zero.	100	Any valid number (see note 1)
SCINC	The increment used to calculate component numbers for STRINGS	1	Any valid number (see note 1)

(continued)

The Macro DGS2K: Keywords Specifiable on Installation			
Keyword	Specifies	Default Value	Alternative Values
SCN	The starting component number for STRINGS	no default	Any **valid** number (see note 1)
SEP	The system separator	'*'	Any **valid** S2K separator, within quotes
TERM	Which terminology is to be output. NEW would **result** in such keywords as SCHEMA-RECORD and CHARACTER, OLD would result in REPEATING-GROUP and NAME being generated	NEW	OLD
UPDATES	Whether the PRODUCE command is' to update and re-encode the S2K-DATABASE member when it generates component numbers	YES	NO

Notes:

1 If component numbers are not specified in a DATAMANAGER S2K-DATABASE definition then when DATAMANAGER PRODUCEs S2K DEFINE Declarations, component numbers will be generated for the appropriate repeating groups, elements, strings and functions. Provision has been made in the macro to declare specific ranges for component number generation but it should be noted that alternative values for component number increments must not result in a number greater than 4095. No default value is specified for starting component numbers for either strings or functions.

2 The value of RNDBIN must be the same as the value of RNDBIN in the macros controlling the generation of Subschema Definitions in COBOL, PL/I and Assembler.